partsim.kdevelop Reference Manual

0.6

Generated by Doxygen 1.3.6

Fri Apr 22 11:06:42 2005

Contents

Chapter 1

partsim.kdevelop Hierarchical Index

1.1 partsim.kdevelop Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

partsim.kdevelop Class Index

2.1 partsim.kdevelop Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

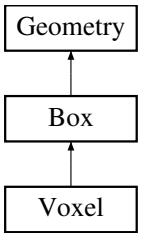
Chapter 3

partsim.kdevelop Class Documentation

3.1 Box Class Reference

`#include <box.h>`

Inheritance diagram for Box::



Public Member Functions

- Box (const Particle::vector_t &p1, const Particle::vector_t &p2)
- virtual bool operator() (const Particle::vector_t &p) const
- const Particle::vector_t & get_corner1 () const
- const Particle::vector_t & get_corner2 () const
- virtual void to_xml (std::ostream &o)

3.1.1 Detailed Description

A box.

Author:
Moritz Franosch

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Box::Box (const Particle::vector_t & *p1*, const Particle::vector_t & *p2*)

Constructs a box with corners p1 and p2.

3.1.3 Member Function Documentation

3.1.3.1 const Particle::vector_t & Box::get_corner1 () const

Returns a corner.

3.1.3.2 const Particle::vector_t & Box::get_corner2 () const

Returns the opposite corner.

3.1.3.3 bool Box::operator() (const Particle::vector_t & *p*) const [virtual]

True if position p is inside of the box.

Implements Geometry.

3.1.3.4 void Box::to_xml (std::ostream & *o*) [virtual]

Writes the box as XML into o.

Implements Geometry.

Reimplemented in Voxel.

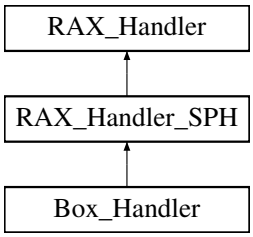
The documentation for this class was generated from the following files:

- box.h
- box.cpp

3.2 Box_Handler Class Reference

`#include <rax_parser_sph.h>`

Inheritance diagram for Box_Handler::



Public Member Functions

- virtual void start (RAX_Parser &parser, const std::string &name, AttributeList &attributes)
- virtual void end (RAX_Parser &parser, const std::string &name)

3.2.1 Detailed Description

Class that handles in .

3.2.2 Member Function Documentation

3.2.2.1 void Box_Handler::end (RAX_Parser & *parser*, const std::string & *name*) [virtual]

Called when ends.

Does nothing.

Reimplemented from RAX_Handler.

3.2.2.2 void Box_Handler::start (RAX_Parser & *parser*, const std::string & *name*, AttributeList & *attributes*) [virtual]

Called when starts.

Reimplemented from RAX_Handler.

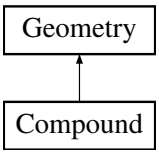
The documentation for this class was generated from the following files:

- rax_parser_sph.h
- rax_parser_sph.cpp

3.3 Compound Class Reference

`#include <compound.h>`

Inheritance diagram for Compound::



Public Member Functions

- Compound ()
- void add (const Pointer< Geometry > g)
- virtual bool operator() (const Particle::vector_t &p) const
- void add_voxel (const Particle::vector_t &p1, const Particle::vector_t &p2)
- void add_pyramid (const Particle::vector_t &top, const Particle::vector_t &c1, const Particle::vector_t &c2, const Particle::vector_t &c3, const Particle::vector_t &c4)
- void reset ()
- bool next ()
- bool at_end () const
- bool empty () const
- int **size** () const
- const Geometry & get_object () const
- Geometry & get_object ()
- virtual void to_xml (std::ostream &o)

3.3.1 Detailed Description

Compound geometric object.

Author:
Moritz Franosch

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Compound::Compound ()

Constructs an empty compound.

3.3.3 Member Function Documentation

3.3.3.1 void Compound::add (const Pointer< Geometry > *g*)

Adds geometric object g.

3.3.3.2 void Compound::add_pyramid (const Particle::vector_t & *top*, const Particle::vector_t & *c1*, const Particle::vector_t & *c2*, const Particle::vector_t & *c3*, const Particle::vector_t & *c4*)

Adds a pyramid with top and a base surface with edges c1, c2, c3 and c4.

3.3.3.3 void Compound::add_voxel (const Particle::vector_t & *p1*, const Particle::vector_t & *p2*)

Adds a voxel with opposite corners p1 and p2.

3.3.3.4 bool Compound::at_end () const

Last next() has returned false.

3.3.3.5 bool Compound::empty () const

Returns true iff compound has no elements.

3.3.3.6 Geometry & Compound::get_object ()

Returns a reference to the current object.

3.3.3.7 const Geometry & Compound::get_object () const

Returns a constant reference to the current object.

3.3.3.8 bool Compound::next ()

Iterator set to next object.

Returns false if no next object exists.

3.3.3.9 bool Compound::operator() (const Particle::vector_t & *p*) const [virtual]

True if position p is inside any of the object that have been added.

Implements Geometry.

3.3.3.10 void Compound::reset ()

Resets iterator through objects.

3.3.3.11 void Compound::to_xml (std::ostream & *o*) [virtual]

Writes the compound as XML into o.

Implements Geometry.

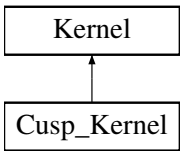
The documentation for this class was generated from the following files:

- compound.h
- compound.cpp

3.4 Cusp_Kernel Class Reference

`#include <cusp_kernel.h>`

Inheritance diagram for Cusp_Kernel::



Public Member Functions

- Cusp_Kernel (const int d, const double h)
- virtual double w (const double x) const
- virtual double dw (const double x) const

3.4.1 Detailed Description

Implements the Cusp kernel W(r) = alpha_d ∗ (1-r)^2 for r<1, 0 else.

Author:
Moritz Franosch

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Cusp_Kernel::Cusp_Kernel (const int *d*, const double *h*)

Constructor, see class Kernel.

3.4.3 Member Function Documentation

3.4.3.1 double Cusp_Kernel::dw (const double *x*) const [virtual]

Returns dw(x)/dx.

Implements Kernel.

3.4.3.2 double Cusp_Kernel::w (const double *x*) const [virtual]

Returns w(x)=(1-x)^2 for x<1, 0 else.

Implements Kernel.

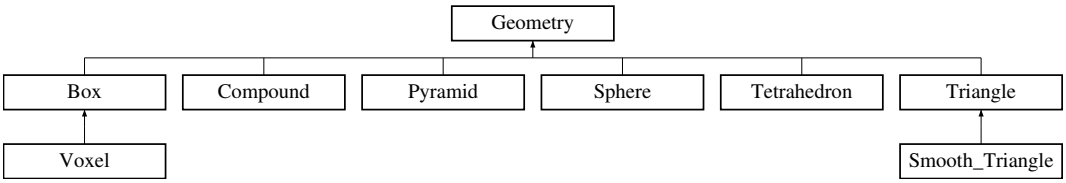
The documentation for this class was generated from the following files:

- cusp_kernel.h
- cusp_kernel.cpp

3.5 Geometry Class Reference

`#include <geometry.h>`

Inheritance diagram for Geometry::



Public Member Functions

- virtual bool operator() (const Particle::vector_t &p) const=0
- virtual void to_xml (std::ostream &o)=0
- virtual ~Geometry ()

3.5.1 Detailed Description

Base class for all geometric objects.

Author:
Moritz Franosch

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Geometry::~Geometry () [virtual]

Virtual destructor.

3.5.3 Member Function Documentation

3.5.3.1 virtual bool Geometry::operator() (const Particle::vector_t & *p*) const [pure virtual]

True if position p is inside the object.

Implemented in Box, Compound, Pyramid, Sphere, Tetrahedron, and Triangle.

3.5.3.2 virtual void Geometry::to_xml (std::ostream & *o*) [pure virtual]

Writes the object as XML into o.

Implemented in Box, Compound, Pyramid, Smooth_Triangle, Sphere, Tetrahedron, Triangle, and Voxel.

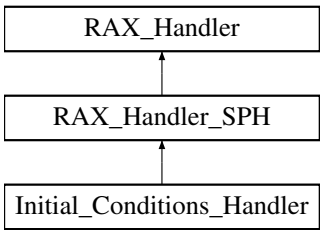
The documentation for this class was generated from the following files:

- geometry.h
- geometry.cpp

3.6 Initial_Conditions_Handler Class Reference

`#include <rax_parser_sph.h>`

Inheritance diagram for Initial_Conditions_Handler::



Public Member Functions

- virtual void start (RAX_Parser &parser, const std::string &name, AttributeList &attributes)
- virtual void end (RAX_Parser &parser, const std::string &name)

3.6.1 Detailed Description

Class that handles .

3.6.2 Member Function Documentation

3.6.2.1 void Initial_Conditions_Handler::end (RAX_Parser & *parser*, const std::string & *name*) [virtual]

Called when ends.

Restores handlers.

Reimplemented from RAX_Handler.

3.6.2.2 void Initial_Conditions_Handler::start (RAX_Parser & *parser*, const std::string & *name*, AttributeList & *attributes*) [virtual]

Called when starts.

Installs handlers for initial conditions.

Reimplemented from RAX_Handler.

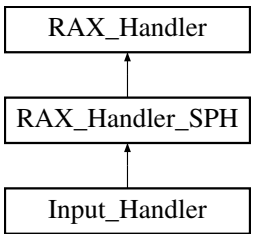
The documentation for this class was generated from the following files:

- rax_parser_sph.h
- rax_parser_sph.cpp

3.7 Input_Handler Class Reference

`#include <rax_parser_sph.h>`

Inheritance diagram for Input_Handler::



Public Member Functions

- virtual void start (RAX_Parser &parser, const std::string &name, AttributeList &attributes)
- virtual void end (RAX_Parser &parser, const std::string &name)
- virtual void text (RAX_Parser &parser, const std::string &name, const std::string &t)

3.7.1 Detailed Description

Class that handles .

3.7.2 Member Function Documentation

3.7.2.1 void Input_Handler::end (RAX_Parser & *parser*, const std::string & *name*) [virtual]

Called when ends.

Prints .

Reimplemented from RAX_Handler.

3.7.2.2 void Input_Handler::start (RAX_Parser & *parser*, const std::string & *name*, AttributeList & *attributes*) [virtual]

Called when starts.

Installs the handlers for input.

Prints .

Reimplemented from RAX_Handler.

3.7.2.3 void Input_Handler::text (RAX_Parser & *parser*, const std::string & *name*, const std::string & *t*) [virtual]

Called when character data is available. (should never be called).

Reimplemented from RAX_Handler.

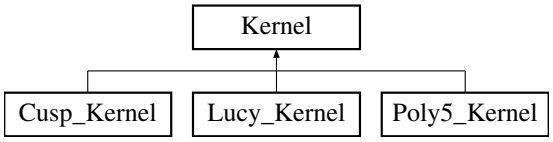
The documentation for this class was generated from the following files:

- rax_parser_sph.h
- rax_parser_sph.cpp

3.8 Kernel Class Reference

`#include <kernel.h>`

Inheritance diagram for Kernel::



Public Member Functions

- Kernel (const int d, const double h, const double alpha_d)
- virtual ~Kernel ()
- double get_h () const
- int get_d () const
- double operator() (const double r) const
- double dW (const double r) const
- const Particle::vector_t grad_r (const Particle::vector_t &r1, const Particle::vector_t &r2) const
- const Particle::vector_t grad_r (const Particle::vector_t &r1, const Particle::vector_t &r2, const double &r12) const
- const Particle::vector_t & grad_r (const Particle::vector_t &r1, const Particle::vector_t &r2, const double &r12, Particle::vector_t &g) const
- virtual double w (const double x) const=0
- virtual double dw (const double x) const=0

Protected Attributes

- int **d**
- double **h**
- double **c_d**

3.8.1 Detailed Description

Smoothing kernel for SPH.

Base class for all kernels.

Author:
Moritz Franosch

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Kernel::Kernel (const int *d*, const double *h*, const double *alpha_d*)

Constructs a kernel for a dimension d and smoothing length h.

The parameter alpha_d is a scaling factor dependent on the dimension.

3.8.2.2 Kernel::~Kernel () [virtual]

Virtual destructor.

3.8.3 Member Function Documentation

3.8.3.1 virtual double Kernel::dw (const double *x*) const [pure virtual]

Returns the derivative dw(x)/dx of the form function w.

Implemented in Cusp_Kernel, Lucy_Kernel, and Poly5_Kernel.

3.8.3.2 double Kernel::dW (const double *r*) const

Returns dW(r)/dr.

3.8.3.3 int Kernel::get_d () const

Returns the dimension of the kernel.

3.8.3.4 double Kernel::get_h () const

Returns the smoothing length (radius) of the kernel.

3.8.3.5 const Particle::vector_t & Kernel::grad_r (const Particle::vector_t & *r1*, const Particle::vector_t & *r2*, const double & *r12*, Particle::vector_t & *g*) const

Writes grad r1 W(|r1-r2|) into vector g and returns a reference to it.

Precondition: r12=|r1-r2|.

3.8.3.6 const Particle::vector_t Kernel::grad_r (const Particle::vector_t & *r1*, const Particle::vector_t & *r2*, const double & *r12*) const

Returns grad r1 W(|r1-r2|).

Precondition: r12=|r1-r2|.

3.8.3.7 const Particle::vector_t Kernel::grad_r (const Particle::vector_t & *r1*, const Particle::vector_t & *r2*) const

Returns grad r1 W(|r1-r2|).

3.8.3.8 double Kernel::operator() (const double *r*) const [inline]

Returns W(r) = alpha_d/h^d ∗ w(r/h).

3.8.3.9 virtual double Kernel::w (const double *x*) const [pure virtual]

Returns the form function w(x).

The precondition int alpha_d w(|r|) dV = 1 must hold.

Implemented in Cusp_Kernel, Lucy_Kernel, and Poly5_Kernel.

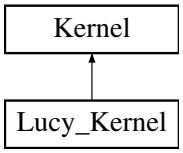
The documentation for this class was generated from the following files:

- kernel.h
- kernel.cpp

3.9 Lucy_Kernel Class Reference

`#include <lucy_kernel.h>`

Inheritance diagram for Lucy_Kernel::



Public Member Functions

- Lucy_Kernel (const int d, const double h)
- virtual double w (const double x) const
- virtual double dw (const double x) const

3.9.1 Detailed Description

Implements the kernel W(r) = alpha_d ∗ (1+3r)(1-r)^3 for r<1, 0 else introduced by Lucy.

Author:
Moritz Franosch

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Lucy_Kernel::Lucy_Kernel (const int *d*, const double *h*)

Constructor, see class Kernel.

3.9.3 Member Function Documentation

3.9.3.1 double Lucy_Kernel::dw (const double *x*) const [virtual]

Returns dw(x)/dx.

Implements Kernel.

3.9.3.2 double Lucy_Kernel::w (const double *x*) const [virtual]

Returns w(x)=(1+3x)(1-x)^3 for x<1, 0 else.

Implements Kernel.

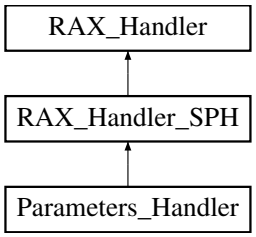
The documentation for this class was generated from the following files:

- lucy_kernel.h
- lucy_kernel.cpp

3.10 Parameters_Handler Class Reference

`#include <rax_parser_sph.h>`

Inheritance diagram for Parameters_Handler::



Public Member Functions

- Parameters_Handler ()
- virtual void start (RAX_Parser &parser, const std::string &name, AttributeList &attributes)
- virtual void end (RAX_Parser &parser, const std::string &name)
- virtual void text (RAX_Parser &parser, const std::string &name, const std::string &t)

3.10.1 Detailed Description

Class that handles .

3.10.2 Constructor & Destructor Documentation

3.10.2.1 Parameters_Handler::Parameters_Handler ()

Constructor.

3.10.3 Member Function Documentation

3.10.3.1 void Parameters_Handler::end (RAX_Parser & *parser*, const std::string & *name*) [virtual]

Called when ends.

Constructs the SPH_Space of class RAX_Parser_SPH with the stored parameters.

Reimplemented from RAX_Handler.

3.10.3.2 void Parameters_Handler::start (RAX_Parser & *parser*, const std::string & *name*, AttributeList & *attributes*) [virtual]

Called when starts.

Stores the parameters.

Reimplemented from RAX_Handler.

3.10.3.3 void Parameters_Handler::text (RAX_Parser & *parser*, const std::string & *name*, const std::string & *t*) [virtual]

Called when character data is available. (should never be called).

Reimplemented from RAX_Handler.

The documentation for this class was generated from the following files:

- rax_parser_sph.h
- rax_parser_sph.cpp

3.11 Particle Class Reference

`#include <particle.h>`

Public Types

- typedef blitz::TinyVector< double, 3 > **vector_t**

Public Member Functions

- Particle (const vector_t &p)
- Particle ()
- const vector_t & **get_position** () const
- const vector_t & **get_velocity** () const
- const vector_t & **get_acceleration** () const
- double **get_mass** () const
- double **get_density** () const
- void **set_position** (const vector_t &p)
- void **set_velocity** (const vector_t &v)
- void **set_acceleration** (const vector_t &a)
- void **set_mass** (const double m)
- void **set_density** (const double rho)
- void **set_drho** (const double drho)
- void update_Euler (const double dt)
- double update_modified_midpoint (const int m, const double dt)

Static Public Member Functions

- std::string string_vector (const Particle::vector_t &v)
- std::string list_vector (const Particle::vector_t &v)

3.11.1 Detailed Description

A particle in the sense of SPH.

Author:
Moritz Franosch

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Particle::Particle (const vector_t & *p*)

Constructs a particle at potition p.

3.11.2.2 Particle::Particle ()

Constructs an invalid particle.

3.11.3 Member Function Documentation

3.11.3.1 string Particle::list_vector (const Particle::vector_t & *v*) [static]

Converts the vector v to a space-separated list v1 v2 ....

3.11.3.2 string Particle::string_vector (const Particle::vector_t & *v*) [static]

Converts the vector v to a string (v1, ...).

3.11.3.3 void Particle::update_Euler (const double *dt*)

Updates the position and the velocity of the particle accoding to its acceleration in one Euler integration step (dt).

3.11.3.4 double Particle::update_modified_midpoint (const int *m*, const double *dt*)

Updates the position and the velocity of the particle according to its acceleration by the modified midpoint method, step m.

The advance in time is returned.

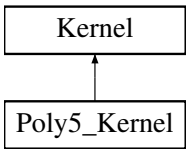
The documentation for this class was generated from the following files:

- particle.h
- particle.cpp

3.12 Poly5_Kernel Class Reference

`#include <poly5_kernel.h>`

Inheritance diagram for Poly5_Kernel::



Public Member Functions

- Poly5_Kernel (const int d, const double h)
- virtual double w (const double x) const
- virtual double dw (const double x) const

3.12.1 Detailed Description

Implements the Poly5 kernel W(r) = alpha_d ∗ (1-r^5)^5 for r<1, 0 else.

Author:
Moritz Franosch

3.12.2 Constructor & Destructor Documentation

3.12.2.1 Poly5_Kernel::Poly5_Kernel (const int *d*, const double *h*)

Constructor, see class Kernel.

3.12.3 Member Function Documentation

3.12.3.1 double Poly5_Kernel::dw (const double *x*) const [virtual]

Returns dw(x)/dx.

Implements Kernel.

3.12.3.2 double Poly5_Kernel::w (const double *x*) const [virtual]

Returns w(x)=(1-x^2)^5 for x<1, 0 else.

Implements Kernel.

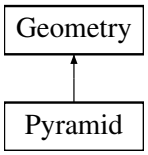
The documentation for this class was generated from the following files:

- poly5_kernel.h
- poly5_kernel.cpp

3.13 Pyramid Class Reference

`#include <pyramid.h>`

Inheritance diagram for Pyramid::



Public Member Functions

- Pyramid (const Particle::vector_t &top, const Particle::vector_t &c1, const Particle::vector_t &c2, const Particle::vector_t &c3, const Particle::vector_t &c4)
- virtual bool operator() (const Particle::vector_t &p) const
- virtual void to_xml (std::ostream &o)

3.13.1 Detailed Description

Pyramid.

Author:
Moritz Franosch

3.13.2 Constructor & Destructor Documentation

3.13.2.1 Pyramid::Pyramid (const Particle::vector_t & *top*, const Particle::vector_t & *c1*, const Particle::vector_t & *c2*, const Particle::vector_t & *c3*, const Particle::vector_t & *c4*)

Constructs a pyramid with top and a base surface with edges c1, c2, c3 and c4.

3.13.3 Member Function Documentation

3.13.3.1 bool Pyramid::operator() (const Particle::vector_t & *p*) const [virtual]

True if position p is inside the object.

Implements Geometry.

3.13.3.2 void Pyramid::to_xml (std::ostream & *o*) [virtual]

Writes the pyramid as XML into o.

Implements Geometry.

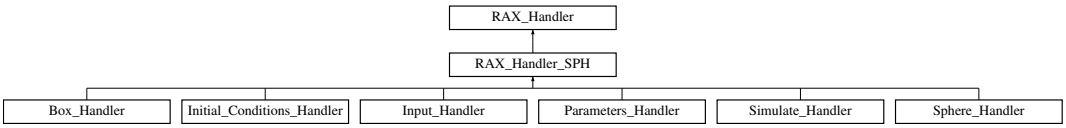
The documentation for this class was generated from the following files:

- pyramid.h
- pyramid.cpp

3.14 RAX_Handler Class Reference

`#include <rax_handler.h>`

Inheritance diagram for RAX_Handler::



Public Member Functions

- RAX_Handler ()
- virtual void start (RAX_Parser &parser, const std::string &name, AttributeList &attributes)
- virtual void end (RAX_Parser &parser, const std::string &name)
- virtual void text (RAX_Parser &parser, const std::string &name, const std::string &t)
- virtual ~RAX_Handler ()
- virtual double double_value (const std::string &s) const
- virtual int int_value (const std::string &s) const

3.14.1 Detailed Description

Handler for class RAX_Parser.

Author:
Moritz Franosch

3.14.2 Constructor & Destructor Documentation

3.14.2.1 RAX_Handler::RAX_Handler ()

Constructor.

3.14.2.2 RAX_Handler::~RAX_Handler () [virtual]

Virtual destructor.

3.14.3 Member Function Documentation

3.14.3.1 double RAX_Handler::double_value (const std::string & *s*) const [virtual]

Converts the string s to a double.

fixme: error handling

3.14.3.2 void RAX_Handler::end (RAX_Parser & *parser*, const std::string & *name*) [virtual]

Called when a tag ends.

Reimplemented in Input_Handler, Parameters_Handler, Initial_Conditions_Handler, Sphere_Handler, Box_Handler, and Simulate_Handler.

3.14.3.3 int RAX_Handler::int_value (const std::string & *s*) const [virtual]

Converts the string s to an int.

fixme: error handling

3.14.3.4 void RAX_Handler::start (RAX_Parser & *parser*, const std::string & *name*, AttributeList & *attributes*) [virtual]

Called when a tag starts.

Reimplemented in Input_Handler, Parameters_Handler, Initial_Conditions_Handler, Sphere_Handler, Box_Handler, and Simulate_Handler.

3.14.3.5 void RAX_Handler::text (RAX_Parser & *parser*, const std::string & *name*, const std::string & *t*) [virtual]

Called before a tag starts or ends when character data is available.

Reimplemented in Input_Handler, Parameters_Handler, and Simulate_Handler.

The documentation for this class was generated from the following files:

- rax_handler.h
- rax_handler.cpp

3.15 RAX_Handler_Pointer Class Reference

`#include <rax_parser.h>`

Public Member Functions

- RAX_Handler_Pointer ()
- RAX_Handler_Pointer (const Pointer< RAX_Handler > handler)

3.15.1 Detailed Description

"Pointer" to a RAX_Handler.

Author:
Moritz Franosch

3.15.2 Constructor & Destructor Documentation

3.15.2.1 RAX_Handler_Pointer::RAX_Handler_Pointer ()

Constructs the RAX_Handler_Pointer with the default handler that just calls RAX_Parser::start_element(), RAX_Parser::end_element(), RAX_Parser::text(), respectively.

3.15.2.2 RAX_Handler_Pointer::RAX_Handler_Pointer (const Pointer< RAX_Handler > *handler*)

Constructs the RAX_Handler_Pointer with this handler.

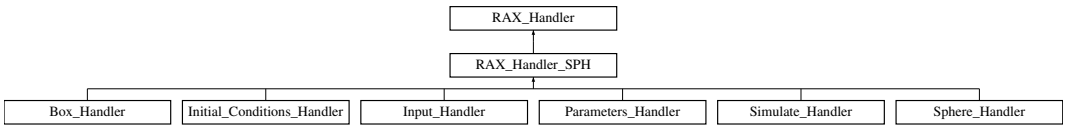
The documentation for this class was generated from the following files:

- rax_parser.h
- rax_parser.cpp

3.16 RAX_Handler_SPH Class Reference

`#include <rax_parser_sph.h>`

Inheritance diagram for RAX_Handler_SPH::



Public Member Functions

- virtual Particle::vector_t vector_value (const std::string &s) const

3.16.1 Detailed Description

RAX Handler for SPH.

3.16.2 Member Function Documentation

3.16.2.1 Particle::vector_t RAX_Handler_SPH::vector_value (const std::string & *s*) const [virtual]

Converts the string s into a vector.

fixme: error handling

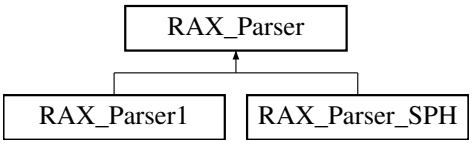
The documentation for this class was generated from the following files:

- rax_parser_sph.h
- rax_parser_sph.cpp

3.17 RAX_Parser Class Reference

`#include <rax_parser.h>`

Inheritance diagram for RAX_Parser::



Public Member Functions

- RAX_Parser ()
- virtual void start_element (const std::string &name, AttributeList &attributes)
- virtual void end_element (const std::string &name)
- virtual void text (const std::string &name, const std::string &t)
- bool parse (const std::string &filename)
- void push_handlers ()
- void pop_handlers ()
- void set_handler (const std::string &element_name, Pointer< RAX_Handler > handler)
- virtual ~RAX_Parser ()

Static Public Member Functions

- const std::string xmlch_to_string (const XMLCh *const name)

3.17.1 Detailed Description

Recursive API for XML.

Author:
Moritz Franosch

3.17.2 Constructor & Destructor Documentation

3.17.2.1 RAX_Parser::RAX_Parser ()

Constructs a RAX Parser.

3.17.2.2 RAX_Parser::~RAX_Parser () [virtual]

Virtual destructor.

3.17.3 Member Function Documentation

3.17.3.1 void RAX_Parser::end_element (const std::string & *name*) [virtual]

Called whenever an end-tag is encountered.

By default, do nothing. Application writers may override this method in a subclass to take specific actions at the end of each element (such as finalising a tree node or writing output to a file).

name The name of the end-tag.

Reimplemented in RAX_Parser_SPH.

3.17.3.2 bool RAX_Parser::parse (const std::string & *filename*)

Parses XML-File filename.

Returns true if successful.

Reimplemented in RAX_Parser_SPH.

3.17.3.3 void RAX_Parser::pop_handlers ()

Pops the last pushed handles from the stack.

3.17.3.4 void RAX_Parser::push_handlers ()

Pushes the current handlers on the stack.

3.17.3.5 void RAX_Parser::set_handler (const std::string & *element_name*, Pointer< RAX_Handler > *handler*)

Sets ∗handler as a handler for element with name element_name.

3.17.3.6 void RAX_Parser::start_element (const std::string & *name*, AttributeList & *attributes*) [virtual]

Called whenever a start-tag is encountered.

By default, do nothing. Application writers may override this method in a subclass to take specific actions at the start of each element (such as allocating a new tree node or writing output to a file).

name The name of the end-tag.

attributes The specified or defaulted attributes.

Reimplemented in RAX_Parser_SPH.

3.17.3.7 void RAX_Parser::text (const std::string & *name*, const std::string & *t*) [virtual]

Called whenever a start- or an end-tag is encountered.

By default, do nothing. Application writers may override this method in a subclass to take specific actions for each chunk of character data.

name The name of the XML-tag in which context the character data was collected.

t The character data.

Reimplemented in RAX_Parser_SPH.

3.17.3.8 const string RAX_Parser::xmlch_to_string (const XMLCh ∗const *name*) [static]

Converts a XMLCh∗ to a std::string.

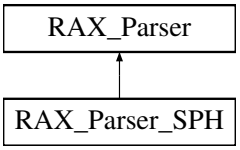
The documentation for this class was generated from the following files:

- rax_parser.h
- rax_parser.cpp

3.18 RAX_Parser_SPH Class Reference

`#include <rax_parser_sph.h>`

Inheritance diagram for RAX_Parser_SPH::



Public Member Functions

- RAX_Parser_SPH ()
- bool parse (const std::string &filename)
- void construct_sph (const int dimensions, const double xdim, const double ydim, const double zdim, const double h, const double rho)
- void set_gravity (const Particle::vector_t &g)
- void set_nu (const double nu)
- void set_c (const double c)
- void set_dt (const double dt)
- bool set_kernel (const std::string &name)
- void set_dt_display (const double dt_display)
- void set_nspheres (const int nspheres)
- void set_nparticles_row (const int nparticles_row)
- void set_dmax (const double d_max_ini)
- void simulate (const double duration)
- SPH & get_sph ()
- void add_sphere (const Particle::vector_t &p, const double r)
- void add_box (const Particle::vector_t &p1, const Particle::vector_t &p2)
- virtual void start_element (const std::string &name, AttributeList &attributes)
- virtual void end_element (const std::string &name)
- virtual void text (const std::string &name, const std::string &t)

3.18.1 Detailed Description

Parser for input files.

Author:
Moritz Franosch

3.18.2 Constructor & Destructor Documentation

3.18.2.1 RAX_Parser_SPH::RAX_Parser_SPH ()

Constructor.

3.18.3 Member Function Documentation

3.18.3.1 void RAX_Parser_SPH::add_box (const Particle::vector_t & *p1*, const Particle::vector_t & *p2*)

Adds a box with one corner at p and the other at p2.

3.18.3.2 void RAX_Parser_SPH::add_sphere (const Particle::vector_t & *p*, const double *r*)

Adds a sphere with radius r at position p.

3.18.3.3 void RAX_Parser_SPH::construct_sph (const int *dimensions*, const double *xdim*, const double *ydim*, const double *zdim*, const double *h*, const double *rho*)

Constructs an sph-space with the given parameters.

3.18.3.4 void RAX_Parser_SPH::end_element (const std::string & *name*) [virtual]

Called whenever an end-tag is encountered and no handler is present (should never be called).

Reimplemented from RAX_Parser.

3.18.3.5 SPH & RAX_Parser_SPH::get_sph ()

Returns the class SPH that is used for the simulation.

construct_sph() was called.

3.18.3.6 bool RAX_Parser_SPH::parse (const std::string & *filename*)

Parses the XML-file filename.

Executes the SPH-simulation.

Returns true if successful.

Reimplemented from RAX_Parser.

3.18.3.7 void RAX_Parser_SPH::set_c (const double *c*)

Sets velocity of sound.

construct_sph() was called.

3.18.3.8 void RAX_Parser_SPH::set_dmax (const double *d_max_ini*)

Sets the maximum length of the triangles when triangulating the surface.

3.18.3.9 void RAX_Parser_SPH::set_dt (const double *dt*)

Sets timestep.

construct_sph() was called.

3.18.3.10 void RAX_Parser_SPH::set_dt_display (const double *dt_display*)

Sets display timestep.

3.18.3.11 void RAX_Parser_SPH::set_gravity (const Particle::vector_t & *g*)

Sets gravity.

construct_sph() was called.

3.18.3.12 bool RAX_Parser_SPH::set_kernel (const std::string & *name*)

Sets the kernel.

Returns true if successful.

construct_sph() was called.

3.18.3.13 void RAX_Parser_SPH::set_nparticles_row (const int *nparticles_row*)

Sets the number of particles in a row.

3.18.3.14 void RAX_Parser_SPH::set_nspheres (const int *nspheres*)

Sets the number of spheres to approximate the fluid.

3.18.3.15 void RAX_Parser_SPH::set_nu (const double *nu*)

Sets viscosity.

construct_sph() was called.

3.18.3.16 void RAX_Parser_SPH::simulate (const double *duration*)

Runs simulation for this duration.

construct_sph() was called.

3.18.3.17 void RAX_Parser_SPH::start_element (const std::string & *name*, AttributeList & *attributes*) [virtual]

Called whenever a start-tag is encountered and no handler is present (should never be called).

Reimplemented from RAX_Parser.

3.18.3.18 void RAX_Parser_SPH::text (const std::string & *name*, const std::string & *t*) [virtual]

Called whenever a start- or an end-tag is encountered and no handler is present (should never be called).

Reimplemented from RAX_Parser.

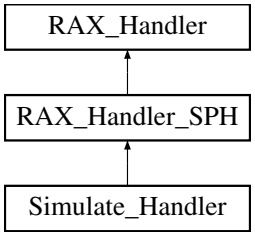
The documentation for this class was generated from the following files:

- rax_parser_sph.h
- rax_parser_sph.cpp

3.19 Simulate_Handler Class Reference

`#include <rax_parser_sph.h>`

Inheritance diagram for Simulate_Handler::



Public Member Functions

- Simulate_Handler ()
- virtual void start (RAX_Parser &parser, const std::string &name, AttributeList &attributes)
- virtual void end (RAX_Parser &parser, const std::string &name)
- virtual void text (RAX_Parser &parser, const std::string &name, const std::string &t)

3.19.1 Detailed Description

Class that handles .

3.19.2 Constructor & Destructor Documentation

3.19.2.1 Simulate_Handler::Simulate_Handler ()

Constructor.

3.19.3 Member Function Documentation

3.19.3.1 void Simulate_Handler::end (RAX_Parser & *parser*, const std::string & *name*) [virtual]

Called when ends.

Reimplemented from RAX_Handler.

3.19.3.2 void Simulate_Handler::start (RAX_Parser & *parser*, const std::string & *name*, AttributeList & *attributes*) [virtual]

Called when starts.

Reimplemented from RAX_Handler.

3.19.3.3 void Simulate_Handler::text (RAX_Parser & *parser*, const std::string & *name*, const std::string & *t*) [virtual]

Called when character data is available. (should never be called).

Reimplemented from RAX_Handler.

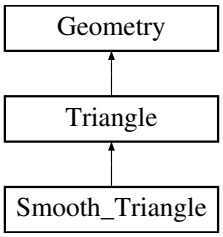
The documentation for this class was generated from the following files:

- rax_parser_sph.h
- rax_parser_sph.cpp

3.20 Smooth_Triangle Class Reference

`#include <smooth_triangle.h>`

Inheritance diagram for Smooth_Triangle::



Public Member Functions

- Smooth_Triangle (const Particle::vector_t &p1, const Particle::vector_t &p2, const Particle::vector_t &p3, const Particle::vector_t &n1, const Particle::vector_t &n2, const Particle::vector_t &n3)
- const Particle::vector_t & get_normal1 () const
- const Particle::vector_t & get_normal2 () const
- const Particle::vector_t & get_normal3 () const
- virtual void to_xml (std::ostream &o)

3.20.1 Detailed Description

A smooth triangle (with normals on each corner).

Author:
Moritz Franosch

3.20.2 Constructor & Destructor Documentation

3.20.2.1 Smooth_Triangle::Smooth_Triangle (const Particle::vector_t & *p1*, const Particle::vector_t & *p2*, const Particle::vector_t & *p3*, const Particle::vector_t & *n1*, const Particle::vector_t & *n2*, const Particle::vector_t & *n3*)

Constructs a triangle with corners p1, p2, p3 and normals n1, n2, n3.

3.20.3 Member Function Documentation

3.20.3.1 const Particle::vector_t & Smooth_Triangle::get_normal1 () const

Returns the normal on the first corner.

3.20.3.2 const Particle::vector_t & Smooth_Triangle::get_normal2 () const

Returns the normal on the second corner.

3.20.3.3 const Particle::vector_t & Smooth_Triangle::get_normal3 () const

Returns the normal on the third corner.

3.20.3.4 void Smooth_Triangle::to_xml (std::ostream & *o*) [virtual]

Writes the triangle as XML into o.

Reimplemented from Triangle.

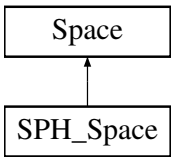
The documentation for this class was generated from the following files:

- smooth_triangle.h
- smooth_triangle.cpp

3.21 Space Class Reference

`#include <space.h>`

Inheritance diagram for Space::



Public Member Functions

- Space (const double length_x, const double length_y, const double length_z, const double diameter)
- double get_diameter () const
- double get_length_x () const
- double get_length_y () const
- double get_length_z () const
- double get_length (const int i) const
- void insert_particle (const Particle::vector_t &p)
- void reset ()
- bool next ()
- bool at_end () const
- const Particle & get_particle () const
- void reset_neighborhood ()
- void reset_neighborhood (double &r_sqr)
- void reset_neighborhood (const Particle::vector_t &p, double &r_sqr)
- bool next_neighbor ()
- bool next_neighbor (double &r_sqr)
- bool at_end_of_neighborhood () const
- const Particle & get_neighbor_particle () const
- void reorder_particles ()

Protected Member Functions

- Particle & get_particle ()
- Particle & get_neighbor_particle ()

3.21.1 Detailed Description

Space to store particles.

Iterate though particles and their neighborhood (i.e. neighboring particles) in an efficient manner.

Author:
Moritz Franosch

3.21.2 Constructor & Destructor Documentation

3.21.2.1 Space::Space (const double *length_x*, const double *length_y*, const double *length_z*, const double *diameter*)

Constructs a space of particles.

The space has dimensions (length_x,length_y,length_z) and diameter of one cell is diameter.

3.21.3 Member Function Documentation

3.21.3.1 bool Space::at_end () const

Last next() has returned false.

3.21.3.2 bool Space::at_end_of_neighborhood () const [inline]

Last next_neighborhood() has returned false.

3.21.3.3 double Space::get_diameter () const

Returns the diameter of a cell.

3.21.3.4 double Space::get_length (const int *i*) const

Returns the length in dimension i of the space.

3.21.3.5 double Space::get_length_x () const

Returns the length in x-dimension of the space.

3.21.3.6 double Space::get_length_y () const

Returns the length in y-dimension of the space.

3.21.3.7 double Space::get_length_z () const

Returns the length in z-dimension of the space.

3.21.3.8 const Particle& Space::get_neighbor_particle () const [inline]

Returns a reference to current particle in the neighborhood of the current particle.

3.21.3.9 Particle& Space::get_neighbor_particle () [inline, protected]

Returns a reference to current particle in the neighborhood of the current particle.

3.21.3.10 const Particle& Space::get_particle () const [inline]

Returns a reference to current particle.

3.21.3.11 Particle& Space::get_particle () [inline, protected]

Returns a reference to current particle.

3.21.3.12 void Space::insert_particle (const Particle::vector_t & *p*)

Insert particle at postion p.

Calls reset().

Invalidates all references to particles.

3.21.3.13 bool Space::next ()

Iterator set to next particle.

Returns false if no next particle exists.

3.21.3.14 bool Space::next_neighbor (double & *r_sqr*) [inline]

Same as next_neighbor(), besides

r_sqr is the square of the euclidian distance between the particle and its neighbor if the return value is true (undefined otherwise).

3.21.3.15 bool Space::next_neighbor () [inline]

Iterator of neighborhood set to next particle.

Returns false if no next particle exists.

3.21.3.16 void Space::reorder_particles ()

Reorders the particles.

Must be called when particles' positions have been changed.

Resets all iterators (see reset()).

3.21.3.17 void Space::reset ()

Resets iterator through particles.

3.21.3.18 void Space::reset_neighborhood (const Particle::vector_t & *p*, double & *r_sqr*)

Resets iterator through neighborhood of current position p.

Invalidates current particle.

3.21.3.19 void Space::reset_neighborhood (double & *r_sqr*)

Same as reset_neighborhood(), besides

r_sqr is the square of the euclidian distance between the particle and its neighbor if !at_end_of_neighborhood() (undefined otherwise).

3.21.3.20 void Space::reset_neighborhood ()

Resets iterator through neighborhood of current particle.

The documentation for this class was generated from the following files:

- space.h
- space.cpp

3.22 SPH Class Reference

`#include <sph.h>`

Public Member Functions

- SPH (const int dimensions, const double xdim, const double ydim, const double zdim, const double h, const double velocity_of_sound, const double dt)
- const SPH_Space & get_space () const
- void set_dt (const double dt)
- void set_density (const double rho)
- void set_viscosity (const double nu)
- void set_gravity (const Particle::vector_t &g)
- void set_velocity_of_sound (const double c)
- void set_boundary_width (const double b)
- void set_boundary_repulsive_force (const double f)
- bool set_kernel (const std::string &name)
- void reset ()
- bool next ()
- bool at_end () const
- const Particle & get_particle () const
- void insert_particle (const Particle::vector_t &p)
- void start ()
- double get_time () const
- double get_dt () const
- double get_h () const
- int get_dimensions () const
- double get_boundary_width () const
- double get_gravity_energy ()
- double get_pressure_energy ()
- double get_kinetic_energy ()
- double get_max_acceleration ()
- double get_density_deviation ()
- void update ()
- bool is_inner_point (const Particle::vector_t &p)
- double get_implicit_surface (const Particle::vector_t &p)
- double get_density (const Particle::vector_t &p)

3.22.1 Detailed Description

Smoothed Particle Hydrodynamics.

Author:
Moritz Franosch

3.22.2 Constructor & Destructor Documentation

3.22.2.1 SPH::SPH (const int *dimensions*, const double *xdim*, const double *ydim*, const double *zdim*, const double *h*, const double *velocity_of_sound*, const double *dt*)

Constructs a SPH space.

3.22.3 Member Function Documentation

3.22.3.1 bool SPH::at_end () const

Last next() has returned false.

3.22.3.2 double SPH::get_boundary_width () const

Returns width of the boundary.

3.22.3.3 double SPH::get_density (const Particle::vector_t & *p*)

Returns the density at position p.

Invalidates iterator through particles.

3.22.3.4 double SPH::get_density_deviation ()

Returns the deviation sqrt(<(rho_i-rho_0)^2>)/rho_0 of the densities.

3.22.3.5 int SPH::get_dimensions () const

Returns the number of dimensions.

3.22.3.6 double SPH::get_dt () const

Returns current timestep in s.

3.22.3.7 double SPH::get_gravity_energy ()

Returns current gravity energy in Joule.

3.22.3.8 double SPH::get_h () const

Returns the radius of the kernel (smoothing length) in m.

3.22.3.9 double SPH::get_implicit_surface (const Particle::vector_t & *p*)

A differentiable function that returns a value >0 iff position p is inside the fluid and <0 otherwise.

3.22.3.10 double SPH::get_kinetic_energy ()

Returns current kinetic energy in Joule.

3.22.3.11 double SPH::get_max_acceleration ()

Returns the maximum acceleration of the particles in m/s^2.

3.22.3.12 const Particle & SPH::get_particle () const

Returns a reference to current particle.

3.22.3.13 double SPH::get_pressure_energy ()

Returns current pressure energy in Joule.

3.22.3.14 const SPH_Space & SPH::get_space () const

Returns a reference to the SPH_Space used.

3.22.3.15 double SPH::get_time () const

Returns current simulated time in s.

3.22.3.16 void SPH::insert_particle (const Particle::vector_t & *p*)

Insert particle at postion p.

Calls reset().

Invalidates all references to particles.

3.22.3.17 bool SPH::is_inner_point (const Particle::vector_t & *p*)

True iff position p is inside the fluid, i.e. iff get_implicit_surface()>0.

Invalidates iterator through particles.

3.22.3.18 bool SPH::next ()

Iterator set to next particle.

Returns false if no next particle exists.

3.22.3.19 void SPH::reset ()

Resets iterator through particles.

3.22.3.20 void SPH::set_boundary_repulsive_force (const double *f*)

Sets repulsive force of the boundary.

3.22.3.21 void SPH::set_boundary_width (const double *b*)

Sets width of the boundary.

3.22.3.22 void SPH::set_density (const double *rho*)

Sets the initial density of the fluid.

3.22.3.23 void SPH::set_dt (const double *dt*)

Sets timestep.

3.22.3.24 void SPH::set_gravity (const Particle::vector_t & *g*)

Sets the gravity acceleration.

3.22.3.25 bool SPH::set_kernel (const std::string & *name*)

Sets the kernel.

Returns true if successful.

3.22.3.26 void SPH::set_velocity_of_sound (const double *c*)

Sets the velocity of sound.

3.22.3.27 void SPH::set_viscosity (const double *nu*)

Sets the viscosity of the fluid.

3.22.3.28 void SPH::start ()

Must be called after all particles are inserted and before the first update().

3.22.3.29 void SPH::update ()

Updates positions and velocities of all particles according to SPH.

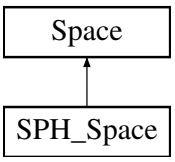
The documentation for this class was generated from the following files:

- sph.h
- sph.cpp

3.23 SPH_Space Class Reference

`#include <sph_space.h>`

Inheritance diagram for SPH_Space::



Public Member Functions

- SPH_Space (const double length_x, const double length_y, const double length_z, const double diameter)
- const Particle::vector_t & get_particle_position () const
- const Particle::vector_t & get_neighbor_position () const
- const Particle::vector_t & get_particle_velocity () const
- const Particle::vector_t & get_neighbor_velocity () const
- double get_particle_mass () const
- double get_neighbor_mass () const
- void set_particle_mass (const double m)
- double get_particle_density () const
- void set_particle_density (const double rho)
- double get_neighbor_density () const
- void set_particle_drho (const double &drho)
- void set_particle_acceleration (const Particle::vector_t &a)
- double update_particles (const double dt)

3.23.1 Detailed Description

Particle space with access functions for SPH.

Author:
Moritz Franosch

3.23.2 Constructor & Destructor Documentation

3.23.2.1 SPH_Space::SPH_Space (const double *length_x*, const double *length_y*, const double *length_z*, const double *diameter*)

Constructs a space of particles.

The space has dimensions (length_x,length_y,length_z) and diameter of one cell is diameter.

3.23.3 Member Function Documentation

3.23.3.1 double SPH_Space::get_neighbor_density () const [inline]

Returns the density of the current neighbor particle.

3.23.3.2 double SPH_Space::get_neighbor_mass () const [inline]

Returns the mass of the current neighbor particle.

3.23.3.3 const Particle::vector_t& SPH_Space::get_neighbor_position () const [inline]

Returns the position of the current neighbor particle.

3.23.3.4 const Particle::vector_t& SPH_Space::get_neighbor_velocity () const [inline]

Returns the velocity of the current neighbor particle.

3.23.3.5 double SPH_Space::get_particle_density () const [inline]

Returns the density of the current particle.

3.23.3.6 double SPH_Space::get_particle_mass () const [inline]

Returns the mass of the current particle.

3.23.3.7 const Particle::vector_t& SPH_Space::get_particle_position () const [inline]

Returns the position of the current particle.

3.23.3.8 const Particle::vector_t& SPH_Space::get_particle_velocity () const [inline]

Returns the velocity of the current particle.

3.23.3.9 void SPH_Space::set_particle_acceleration (const Particle::vector_t & *a*)

Set the acceleration of the current particle.

3.23.3.10 void SPH_Space::set_particle_density (const double *rho*)

Sets the density of the current particle to rho.

3.23.3.11 void SPH_Space::set_particle_drho (const double & *drho*)

Set the drho/dt of the current particle.

3.23.3.12 void SPH_Space::set_particle_mass (const double *m*)

Sets the mass of the current particle to m.

3.23.3.13 double SPH_Space::update_particles (const double *dt*)

Velocity and positons of particles is updated according to the current accelerations.

Ensures boundary conditions are met.

The advance in time is returned.

Also reorganizes particle layout.

Invalidates all references to particles.

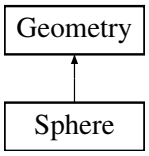
The documentation for this class was generated from the following files:

- sph_space.h
- sph_space.cpp

3.24 Sphere Class Reference

`#include <sphere.h>`

Inheritance diagram for Sphere::



Public Member Functions

- Sphere (const Particle::vector_t &p, const double r)
- void set_radius (const double r)
- const Particle::vector_t & get_position () const
- double get_radius () const
- virtual bool operator() (const Particle::vector_t &p) const
- virtual void to_xml (std::ostream &o)

3.24.1 Detailed Description

A sphere.

Author:
Moritz Franosch

3.24.2 Constructor & Destructor Documentation

3.24.2.1 Sphere::Sphere (const Particle::vector_t & *p*, const double *r*)

Constructs a sphere with p and radius r.

3.24.3 Member Function Documentation

3.24.3.1 const Particle::vector_t & Sphere::get_position () const

Returns the position of the sphere.

3.24.3.2 double Sphere::get_radius () const

Returns the radius of the sphere.

3.24.3.3 bool Sphere::operator() (const Particle::vector_t & *p*) const [virtual]

True if position p is inside the object.

Implements Geometry.

3.24.3.4 void Sphere::set_radius (const double *r*)

Sets the radius of the spere.

3.24.3.5 void Sphere::to_xml (std::ostream & *o*) [virtual]

Writes the sphere as XML into o.

Implements Geometry.

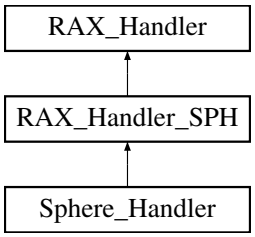
The documentation for this class was generated from the following files:

- sphere.h
- sphere.cpp

3.25 Sphere_Handler Class Reference

`#include <rax_parser_sph.h>`

Inheritance diagram for Sphere_Handler::



Public Member Functions

- virtual void start (RAX_Parser &parser, const std::string &name, AttributeList &attributes)
- virtual void end (RAX_Parser &parser, const std::string &name)

3.25.1 Detailed Description

Class that handles in .

3.25.2 Member Function Documentation

3.25.2.1 void Sphere_Handler::end (RAX_Parser & *parser*, const std::string & *name*) [virtual]

Called when ends.

Does nothing.

Reimplemented from RAX_Handler.

3.25.2.2 void Sphere_Handler::start (RAX_Parser & *parser*, const std::string & *name*, AttributeList & *attributes*) [virtual]

Called when starts.

Reimplemented from RAX_Handler.

The documentation for this class was generated from the following files:

- rax_parser_sph.h
- rax_parser_sph.cpp

3.26 Sphere_Space Class Reference

`#include <sphere_space.h>`

Public Member Functions

- Sphere_Space (SPH &sph, const int nspheres)
- void reset ()
- bool next ()
- bool at_end () const
- const Sphere & get_sphere () const
- void to_xml (std::ostream &o)

3.26.1 Detailed Description

Approximates a 3D SPH structure with spheres.

Author:
Moritz Franosch

3.26.2 Constructor & Destructor Documentation

3.26.2.1 Sphere_Space::Sphere_Space (SPH & *sph*, const int *nspheres*)

Constructs a sphere space from a sph space.

nspheres Number of spheres to construct.

3.26.3 Member Function Documentation

3.26.3.1 bool Sphere_Space::at_end () const

Last next() has returned false.

3.26.3.2 const Sphere & Sphere_Space::get_sphere () const

Returns a reference to the current sphere.

3.26.3.3 bool Sphere_Space::next ()

Iterator set to next sphere.

Returns false if no next sphere exists.

3.26.3.4 void Sphere_Space::reset ()

Resets iterator through spheres.

3.26.3.5 void Sphere_Space::to_xml (std::ostream & *o*)

Output as XML into o.

The documentation for this class was generated from the following files:

- sphere_space.h
- sphere_space.cpp

3.27 State_Equation Class Reference

`#include <state_equation.h>`

Public Member Functions

- State_Equation (const double velocity_of_sound, const double rho_0)
- double get_pressure_from_density (const double rho) const
- double get_energy (const double m, const double rho) const
- void set_rho_0 (const double rho_0)
- void set_velocity_of_sound (const double c)
- double get_rho_0 () const
- double get_velocity_of_sound () const

3.27.1 Detailed Description

Equation of state.

Author:
Moritz Franosch

3.27.2 Constructor & Destructor Documentation

3.27.2.1 State_Equation::State_Equation (const double *velocity_of_sound*, const double *rho_0*)

Constructor.

3.27.3 Member Function Documentation

3.27.3.1 double State_Equation::get_energy (const double *m*, const double *rho*) const

Get pressure from mass m and density rho.

3.27.3.2 double State_Equation::get_pressure_from_density (const double *rho*) const

Get pressure from density.

3.27.3.3 double State_Equation::get_rho_0 () const

Get equilibrium density.

3.27.3.4 double State_Equation::get_velocity_of_sound () const

Get velocity of sound.

3.27.3.5 void State_Equation::set_rho_0 (const double *rho_0*)

Set equilibrium density.

3.27.3.6 void State_Equation::set_velocity_of_sound (const double *c*)

Set velocity of sound.

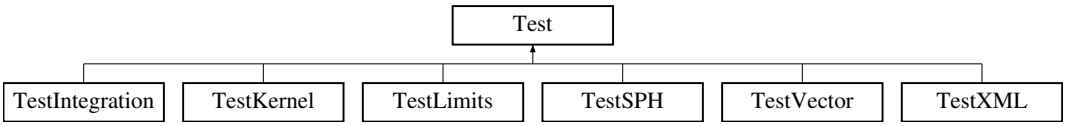
The documentation for this class was generated from the following files:

- state_equation.h
- state_equation.cpp

3.28 Test Class Reference

`#include <test.h>`

Inheritance diagram for Test::



Public Member Functions

- virtual int operator() ()=0

3.28.1 Detailed Description

Base class for all tests.

Author:
Moritz Franosch

3.28.2 Member Function Documentation

3.28.2.1 virtual int Test::operator() () [pure virtual]

Returns EXIT_SUCCESS if successful, EXIT_FAILURE else.

Implemented in TestIntegration, TestKernel, TestLimits, TestSPH, TestVector, and TestXML.

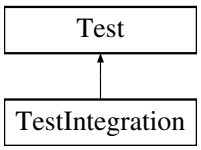
The documentation for this class was generated from the following file:

- test.h

3.29 TestIntegration Class Reference

`#include <testintegration.h>`

Inheritance diagram for TestIntegration::



Public Member Functions

- int operator() ()

3.29.1 Detailed Description

Tests integration routinges of class Particle.

Author:
Moritz Franosch

3.29.2 Member Function Documentation

3.29.2.1 int TestIntegration::operator() () [virtual]

Returns EXIT_SUCCESS if successful, EXIT_FAILURE else.

Implements Test.

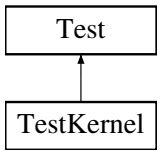
The documentation for this class was generated from the following files:

- testintegration.h
- testintegration.cpp

3.30 TestKernel Class Reference

`#include <testkernel.h>`

Inheritance diagram for TestKernel::



Public Member Functions

- virtual int operator() ()

3.30.1 Detailed Description

Tests SPH kernels.

Author:
Moritz Franosch

3.30.2 Member Function Documentation

3.30.2.1 int TestKernel::operator() () `[virtual]`

Returns EXIT_SUCCESS if successful, EXIT_FAILURE else.

Implements Test.

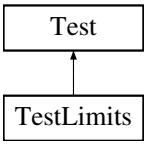
The documentation for this class was generated from the following files:

- testkernel.h
- testkernel.cpp

3.31 TestLimits Class Reference

`#include <testlimits.h>`

Inheritance diagram for TestLimits::



Public Member Functions

- virtual int operator() ()

3.31.1 Detailed Description

Tests numeric limits.

Author:
Moritz Franosch

3.31.2 Member Function Documentation

3.31.2.1 int TestLimits::operator() () [virtual]

Returns EXIT_SUCCESS if successful, EXIT_FAILURE else.

Implements Test.

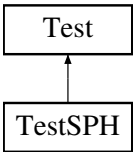
The documentation for this class was generated from the following files:

- testlimits.h
- testlimits.cpp

3.32 TestSPH Class Reference

`#include <testsph.h>`

Inheritance diagram for TestSPH::



Public Member Functions

- virtual int operator() ()

3.32.1 Detailed Description

Tests the class SPH.

Author:
Moritz Franosch

3.32.2 Member Function Documentation

3.32.2.1 int TestSPH::operator() () [virtual]

Returns EXIT_SUCCESS if successful, EXIT_FAILURE else.

Implements Test.

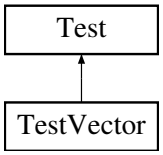
The documentation for this class was generated from the following files:

- testsph.h
- testsph.cpp

3.33 TestVector Class Reference

`#include <testvector.h>`

Inheritance diagram for TestVector::



Public Member Functions

- virtual int operator() ()

3.33.1 Detailed Description

Tests Blitz++ vector.

Author:
Moritz Franosch

3.33.2 Member Function Documentation

3.33.2.1 int TestVector::operator() () [virtual]

Returns EXIT_SUCCESS if successful, EXIT_FAILURE else.

Implements Test.

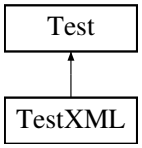
The documentation for this class was generated from the following files:

- testvector.h
- testvector.cpp

3.34 TestXML Class Reference

`#include <testxml.h>`

Inheritance diagram for TestXML::



Public Member Functions

- virtual int operator() ()

3.34.1 Detailed Description

Tests class RAX_Parser.

Author:
Moritz Franosch

3.34.2 Member Function Documentation

3.34.2.1 int TestXML::operator() () [virtual]

Returns EXIT_SUCCESS if successful, EXIT_FAILURE else.

Implements Test.

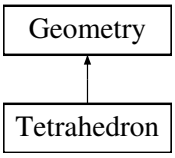
The documentation for this class was generated from the following files:

- testxml.h
- testxml.cpp

3.35 Tetrahedron Class Reference

`#include <tetrahedron.h>`

Inheritance diagram for Tetrahedron::



Public Member Functions

- Tetrahedron (const Particle::vector_t &p1, const Particle::vector_t &p2, const Particle::vector_t &p3, const Particle::vector_t &p4)
- virtual bool operator() (const Particle::vector_t &p) const
- const Particle::vector_t & get_corner1 () const
- const Particle::vector_t & get_corner2 () const
- const Particle::vector_t & get_corner3 () const
- const Particle::vector_t & get_corner4 () const
- virtual void to_xml (std::ostream &o)

3.35.1 Detailed Description

A tetrahedron.

Author:
Moritz Franosch

3.35.2 Constructor & Destructor Documentation

3.35.2.1 Tetrahedron::Tetrahedron (const Particle::vector_t & *p1*, const Particle::vector_t & *p2*, const Particle::vector_t & *p3*, const Particle::vector_t & *p4*)

Constructs a tetrahedron with corners p1, p2, p3 and p4.

3.35.3 Member Function Documentation

3.35.3.1 const Particle::vector_t& Tetrahedron::get_corner1 () const

Returns the first corner.

3.35.3.2 const Particle::vector_t& Tetrahedron::get_corner2 () const

Returns the second corner.

3.35.3.3 const Particle::vector_t& Tetrahedron::get_corner3 () const

Returns the third corner.

3.35.3.4 const Particle::vector_t& Tetrahedron::get_corner4 () const

Returns the fourth corner.

3.35.3.5 virtual bool Tetrahedron::operator() (const Particle::vector_t & *p*) const [virtual]

Not implemented.

Implements Geometry.

3.35.3.6 virtual void Tetrahedron::to_xml (std::ostream & *o*) [virtual]

Writes the tetrahedron as XML into o.

Implements Geometry.

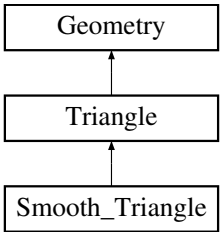
The documentation for this class was generated from the following file:

- tetrahedron.h

3.36 Triangle Class Reference

`#include <triangle.h>`

Inheritance diagram for Triangle::



Public Member Functions

- Triangle (const Particle::vector_t &p1, const Particle::vector_t &p2, const Particle::vector_t &p3)
- virtual bool operator() (const Particle::vector_t &p) const
- const Particle::vector_t & get_corner1 () const
- const Particle::vector_t & get_corner2 () const
- const Particle::vector_t & get_corner3 () const
- virtual void to_xml (std::ostream &o)

3.36.1 Detailed Description

A triangle.

Author:
Moritz Franosch

3.36.2 Constructor & Destructor Documentation

3.36.2.1 Triangle::Triangle (const Particle::vector_t & *p1*, const Particle::vector_t & *p2*, const Particle::vector_t & *p3*)

Constructs a triangle with corners p1, p2 and p3.

3.36.3 Member Function Documentation

3.36.3.1 const Particle::vector_t & Triangle::get_corner1 () const

Returns the first corner.

3.36.3.2 const Particle::vector_t & Triangle::get_corner2 () const

Returns the second corner.

3.36.3.3 const Particle::vector_t & Triangle::get_corner3 () const

Returns the third corner.

3.36.3.4 bool Triangle::operator() (const Particle::vector_t & *p*) const [virtual]

Returns false.

Implements Geometry.

3.36.3.5 void Triangle::to_xml (std::ostream & *o*) [virtual]

Writes the triangle as XML into o.

Implements Geometry.

Reimplemented in Smooth_Triangle.

The documentation for this class was generated from the following files:

- triangle.h
- triangle.cpp

3.37 Triangulator Class Reference

`#include <triangulator.h>`

Public Member Functions

- Triangulator (SPH &sph, const double d_min)
- void reset ()
- bool next ()
- bool at_end () const
- const Geometry & get_triangle () const
- void to_xml (std::ostream &o)

3.37.1 Detailed Description

Triangulates the surfaces of a sph space.

Uses an adoption of the marching tetrahedra algorithm.

Author:
Moritz Franosch

3.37.2 Constructor & Destructor Documentation

3.37.2.1 Triangulator::Triangulator (SPH & *sph*, const double *d_min*)

Constructs triangulated surfaces from a sph space.

d_min Approximate length of the sides of the triangles.

3.37.3 Member Function Documentation

3.37.3.1 bool Triangulator::at_end () const

Last next() has returned false.

3.37.3.2 const Geometry & Triangulator::get_triangle () const

Returns a reference to the current triangle.

3.37.3.3 bool Triangulator::next ()

Iterator set to next triangle.

Returns false if no next triangle exists.

3.37.3.4 void Triangulator::reset ()

Resets iterator through triangles.

3.37.3.5 void Triangulator::to_xml (std::ostream & *o*)

Output of triangulator as XML.

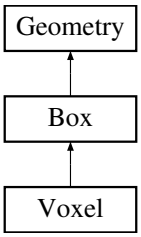
The documentation for this class was generated from the following files:

- triangulator.h
- triangulator.cpp

3.38 Voxel Class Reference

`#include <voxel.h>`

Inheritance diagram for Voxel::



Public Member Functions

- Voxel (const Particle::vector_t &p1, const Particle::vector_t &p2)
- virtual void to_xml (std::ostream &o)

3.38.1 Detailed Description

Voxel.

Author:
Moritz Franosch

3.38.2 Constructor & Destructor Documentation

3.38.2.1 Voxel::Voxel (const Particle::vector_t & *p1*, const Particle::vector_t & *p2*)

Constructs a voxel with corners p1 and p2.

3.38.3 Member Function Documentation

3.38.3.1 void Voxel::to_xml (std::ostream & *o*) `[virtual]`

Writes the voxel as XML into o.

Reimplemented from Box.

The documentation for this class was generated from the following files:

- voxel.h
- voxel.cpp

3.39 Voxelizer Class Reference

`#include <voxelizer.h>`

Public Member Functions

- Voxelizer (SPH &sph, const double d_min)
- void reset ()
- bool next ()
- bool at_end () const
- const Geometry & get_voxel () const
- void to_xml (std::ostream &o)

3.39.1 Detailed Description

Vozelizes a sph space.

Author:
Moritz Franosch

3.39.2 Constructor & Destructor Documentation

3.39.2.1 Voxelizer::Voxelizer (SPH & *sph*, const double *d_min*)

Constructs a voxelized space from a sph space.

The voxels start at diameter sph.get_d() and are recursively made smaller until d_min.

d_min Minimum diameter of voxels.

3.39.3 Member Function Documentation

3.39.3.1 bool Voxelizer::at_end () const

Last next() has returned false.

3.39.3.2 const Geometry & Voxelizer::get_voxel () const

Returns a reference to the current voxel.

3.39.3.3 bool Voxelizer::next ()

Iterator set to next voxel.

Returns false if no next voxel exists.

3.39.3.4 void Voxelizer::reset ()

Resets iterator through voxels.

3.39.3.5 void Voxelizer::to_xml (std::ostream & *o*)

Output of voxelizer as XML.

The documentation for this class was generated from the following files:

- voxelizer.h
- voxelizer.cpp

Index